codemd - Code markdown for authoring

Srini, rs@toprllc.com

September 20, 2020

1 Introduction

This projectlet targets authors of blogs, articles and books myself included. When I like to illustrate a point, I like to show a segment of code and develop accompanying explanatory text. The annotation usually benefits from a line number reference of the example. While we can always extract the source code into the document, it becomes hard to track the original source.

Given sourcefile marked up with special commands, this tool is designed to generate graphic files corresponding to the lines of source code. The special commands are simple enough that including such directives is language agnostic - so it can be used with any programming language. Examples in **C++** and **Ada** included.

1.1 Learning Objectives

Graphic file generation from text Generating a graphic file with primarily text is not too complicated yet will benefit from a suitable library. One of the interesting challenges in any such system is the coordinate system. For each line, the location of the line number, text etc need to be computed by the application. The application can also support a variety of **font**s potentially configurable on the command line.

2 Specifications

A command line tool, this should enable generating graphics from a directory structure of source files and provide an option to add line numbers in the graphic file. Graphic files in the **png** format should be generated named for the original file and the fragment number within a source file.

2.1 Example directives - Markdown of the sources

A sample configuration for a project file is listed below.

Listing 1: Example Source file in C

```
#include <stdio.h>
#include <string>
// codemd: begin
int main(int argc, char **argv) {
   int i ;
```

```
    for (i=1; i<argc; i++) {
        if (argv[i][0] == '-') {
            printf("Found switch %s\n",argv[i]);
        }
    }
}
// codemd: end
```

Listing 2: Example Source file in Ada

```
-- codemd: begin caption="Preliminaries"
with Text_Io; use Text_Io ;
with Ada.Command_Line ; use Text_Io ;
-- codemd: end

-- codemd: begin caption="Main procedure"
procedure switch is
begin
    for arg in 1..Argument_Count
    loop
        if Argument(arg)(1..1) == '-'              -- codemd: callout
        then
           put("Found a Switch ");
           put_line(Argument(arg));                -- codemd: callout
    end loop ;
end switch ;
-- codemd: end
```

2.2 codemd directives - described

As a file is scanned line by line, codemd searches for directives. When a line contains the keyword **codemd:** the line is searched for directives and options. Directives and options are case sensitive.

Since codemd ignores all other markers, it is language agnostic. Directives are included in line comments appropriate for the language (Eg. # for **Python** and **–** for **Ada**).

begin Starts an illustrative section. Creates a graphic file. If a graphic file is currently open - implicitly closes it. Options for this directive are:

- caption="string" provides a name for the caption. This option of the begin directive places the caption above the listing.

end Ends the illustrative section. Closes the graphic file created earlier. There is an implicit end at the end of the file. The options for this directive are:

- caption="string" provides a name for the caption. This option of the end directive places the caption below the listing.

callout

skip Very long procedures with a lot of details not relevant to a discussion could be eliminated from the graphic by using the skip directive. Any code fragments between **skip** and **skipend** will not be emitted to the graphic replaced by an ellipsis.

skipend terminates a skip block.

break this directive causes the currently active graphic file to close and a new one to open; numbered serially. Helps break long files into multiple graphic files for illustration. Line numbers if effective continue. If there is a caption in effect from the begin directive, the caption is repeated at the top of the new graphic file.

3 Implementation

3.1 Libraries

- pngwriter library from `https://github.com/pngwriter/pngwriter.git`
- boost library - `https://www.boost.org/`

Implementation in **C++** `https://gitlab.com/cpp8/codemd.git`